Marco Kunze (makunze@cs.tu-berlin.de) **SS 2002**
Sebastian Nowozin (nowozin@cs.tu-berlin.de) **13.5.2002**

Lösungen zum 3. Übungsblatt TheGI2

Lösung zu Aufgabe 1:

(a) Eine mögliche Untersignatur $Keller'$ zur Signatur $Keller$.

<u>Keller'</u> =	*sorts*	Data, Stack
	opns	d1, d2: → Data
		empty: → Stack
		push: Data Stack → Stack
		top: Stack → Data

(b) Das $Keller'$ Redukt von A.

	$A' := A/_{Keller'}$
Data	$A'_{Data} = \{0, \cdots, 9\}$
Stack	$A'_{Stack} = \{0, \cdots, 9\}^*$
d1	$d1_{A'} \in A'_{Data}$ 0
d2	$d2_{A'} \in A'_{Data}$ 1
empty	$empty_{A'} \in A'_{Stack}$ λ
push	$push_{A'} : A'_{Data} \times A'_{Stack} \to A'_{Stack}$ $(y, v) \mapsto vy$
top	$top_{A'} : A'_{Stack} \to A'_{Data}$ $v \mapsto \begin{cases} 0 & v = \lambda \\ y & v = uy \end{cases}$

(c) C_{Data} muss mindestens ein Element enthalten, da Konstantensymbole für C_{Data} definiert sind. C_{Stack} könnte leer sein, wenn es nur als Teil kartesischer Produkte verwendet werden würde, die Funktionsparameter sind. Damit werden zwar die Abbildungen der Operationen leer, sie erfüllen jedoch immer noch den Syntax der Signatur und bilden damit eine Algebra. Da C_{Stack} jedoch auch durch das Konstantensymbol *empty* benutzt wird, muss es auch mindestens ein Element enthalten.

	C
Data	$C_{Data} = \{0\}$
Stack	$C_{Stack} = \{\lambda\}$
d1	$d1_C \in C_{Data}$ 0
d2	$d2_C \in C_{Data}$ 0
empty	$empty_C \in C_{Stack}$ λ
push	$push_C : C_{Data} \times C_{Stack} \to C_{Stack}$ $(y, v) \mapsto \lambda$
pop	$pop_C : C_{Stack} \to C_{Stack}$ $v \mapsto \lambda$
clean	$clean_C : C_{Stack} \to C_{Stack}$ $v \mapsto \lambda$
top	$top_C : C_{Stack} \to C_{Data}$ $v \mapsto 0$

Lösung zu Aufgabe 2:

(a) Ein solcher Homomorphismus $h : B \to A$ existiert.

s	$(h_s : B_s \to A_s)_{s \in S}$
Data	$h_{Data} = \{0 \mapsto 0, 1 \mapsto 1, \cdots, 9 \mapsto 9\}$
Stack	$h_{Stack} = \{(w, v) \mapsto w\}$

Beweise:

- Konstantensymbole

$$h_{Data}(d1_B) = 0 = d1_A$$

$$h_{Data}(d2_B) = 1 = d2_A$$

$$h_{Stack}(empty_B) = \lambda = empty_A$$

- Operationen
  - push
    $\forall y \in B_{Data}, (w, v) \in B_{Stack}$:

$$h_{Stack}(push_B(y, (w, v))) = \begin{cases} h_{Stack}(wy, \lambda) = wy & \text{für } v = \lambda \\ h_{Stack}(wy, u) = wy & \text{für } v = xu \end{cases}$$

$$push_A(h_{Data}(y), h_{Stack}(w, v)) = push_A(y, w) = wy$$

    Damit ist die Homomorphie bezüglich der Operation *push* bewiesen.
  - pop
    $\forall (w, v) \in B_{Stack}$:

$$h_{Stack}(pop_B(w, v)) = \begin{cases} h_{Stack}(\lambda, v) = \lambda & \text{für } w = \lambda \\ h_{Stack}(u, yv) = u & \text{für } w = uy \end{cases}$$

$$pop_A(h_{Stack}(w, v)) = pop_A(w) = \begin{cases} \lambda & \text{für } w = \lambda \\ u & \text{für } w = uy \end{cases}$$

    Damit ist die Homomorphie bezüglich der Operation *pop* bewiesen.

– clean
$\forall(w,v) \in B_{Stack}$:

$$h_{Stack}(clean_B(w,v)) = h_{Stack}(w,\lambda) = w$$

$$clean_A(h_{Stack}(w,v)) = clean_A(w) = w$$

Damit ist die Homomorphie bezüglich der Operation *clean* bewiesen.

– top
$\forall(w,v) \in B_{Stack}$:

$$h_{Data}(top_B(w,v)) = \begin{cases} h_{Data}(0) = 0 & \text{für } w = \lambda \\ h_{Data}(y) = y & \text{für } w = uy \end{cases}$$

$$top_A(h_{Data}(w,v)) = top_A(w) = \begin{cases} 0 & \text{für } w = \lambda \\ y & \text{für } w = uy \end{cases}$$

(b) Behauptung: Es gibt keinen Homomorphismus h mit $h : A \to B$.

Beweis: Beweis durch Widerspruch. Wir nehmen an es existiert ein Homomorphismus h. Da h_{Stack} rechtseindeutig und linkstotal sein muss, aber auf ein 2-Tupel abbildet, muss h_{Stack} zwingend von der Form sein:

$$h_{Stack} = \{w \mapsto (f_1(w), f_2(w))\}$$

Dabei ist zwangsläufig $(f_1(w), f_2(w)) \in B_{Stack}$. Dann muss aber gelten:

$$h_{Stack}(pop_A(v)) = \begin{cases} h_{Stack}(\lambda) = (f_1(\lambda), f_2(\lambda)) & \text{für } v = \lambda \\ h_{Stack}(u) = (f_1(u), f_2(u)) & \text{für } v = uy \end{cases}$$

und

$$pop_B(h_{Stack}(v)) = pop_B(f_1(v), f_2(v)) = \begin{cases} (\lambda, f_2(v)) & \text{für } f_1(v) = \lambda \\ (u, yf_2(v)) & \text{für } f_1(v) = uy \end{cases}$$

Man kann jetzt für $v = \lambda$ zeigen, dass $f_1(\lambda) = \lambda$ sein muss. Die Beweisidee war, den Widerspruch durch Verletzen der Rechtseindeutigkeit zu erzeugen, also zwei Abbildungen in h_{Stack} zu finden, so dass zum Beispiel $(\lambda \mapsto (\lambda, f_2(\lambda)))$ und $(\lambda \mapsto (\lambda, yf_2(\lambda)))$ existieren müssten. Leider ist das nicht ganz geglückt.